

Bsmooth: Learning from user feedback to disambiguate query terms in interactive data retrieval

Bernardo Gonçalves
IBM Research – São Paulo, Brazil
begoncalves@acm.org

H. V. Jagadish
Univ. of Michigan – Ann Arbor, USA
jag@umich.edu

Abstract—There is great interest in supporting imprecise queries over databases today. To support such queries, the system is typically required to disambiguate parts of the user-specified query against the database, using whatever resources are intrinsically available to it (the database schema, value distributions, natural language models etc). Often, systems will also have a user-interaction log available, which can supplement their model based on their own intrinsic resources. This leads to a problem of how best to combine the system’s prior ranking with insight derived from the user-interaction log. Statistical inference techniques such as maximum likelihood or Bayesian updates from a subjective prior turn out not to apply in a straightforward way due to possible noise from user search behavior and to encoding biases endemic to the system’s models. In this paper, we address such learning problems in interactive data retrieval, with specific focus on type classification for user-specified query terms. We develop a novel Bayesian smoothing algorithm, **Bsmooth**, which is simple, fast, flexible and accurate. We analytically establish some desirable properties and show, through experiments against an independent benchmark, that the addition of such a learning layer performs much better than standard methods.

I. INTRODUCTION

The last decade has seen significant research activity on imprecise query processing over structured data. Non-expert end-users are accessing databases, often over the web, and posing queries as keyword collections, natural language sentences, fielded values with some partial structure, and even as clicks on a GUI. The system then bears the responsibility of disambiguating the given imprecise query onto a complete, precise query that best reflects the user’s intent.

The typical state-of-the-art approach to the problem is based on scoring functions that synthesize various resources such as the database schema, value distributions, natural language models etc, which are all intrinsic to the database. If user feedback can be captured from previous sessions, it should be possible to use it as an extrinsic resource to improve upon the system’s intrinsic interpretation, as we see in this example.

Example 1: Consider query Q , alternatively specified in keyword and natural language against a relational store of the Internet Movie Database (IMDb) in its original design.

Q . ‘*tom hanks*’ 2004.

Q' . find all movies played by ‘*tom hanks*’ in 2004.

For term ‘*tom hanks*,’ this is the top-5 ranking of disambiguation options (in IMDb’s *TABLE.attribute* design) according to some scoring function intrinsic to the database.¹

- | | |
|-------------------|----------|
| 1. CHAR_NAME.name | (27.93%) |
| 2. NAME.name | (23.46%) |

- | | |
|--------------------|----------|
| 3. TITLE.title | (22.07%) |
| 4. MOVIE_INFO.info | (15.08%) |
| 5. ROLE_TYPE.role | (11.45%) |

The system is unsure about whether ‘*tom hanks*’ is the name of a character, name of a person, or a movie title. The desired answer is NAME.name, which is ranked second. Based on information intrinsic to the database including a language model, this is the best the system has been able to do. □

By learning from user feedback provided interactively over some sessions, we consider whether the system in this example may have been able to do better. Specifically, we consider the following scenario: Query answer takes place in two steps: (S1) the imprecise query is disambiguated onto existing database elements, and then (S2) a precise SQL query with possible join operations is generated and processed to return with the answer. The session is optionally interactive after S1 to allow the user to correct the interpretation of the query, before it is actually evaluated in S2.

Example 1: (continued). For the same term ‘*tom hanks*,’ this is an instance of a simulated user feedback log collected from a crowdsourced experiment that we describe later. Such an information source is considered extrinsic to the database.

- | | |
|--------------------|-----|
| 1. CHAR_NAME.name | (1) |
| 2. NAME.name | (8) |
| 3. TITLE.title | (0) |
| 4. MOVIE_INFO.info | (1) |
| 5. ROLE_TYPE.role | (0) |

We find that 8 out of 10 users in this interaction log think ‘*tom hanks*’ is the name of a person, but the remaining 2 users prefer other choices. Our problem is to combine this user feedback with the intrinsic system scores shown above, which suggest character name as the most likely interpretation. □

Our inference problem seems to suit bayesian learning: the intrinsic model is a prior distribution, and every user choice is an observation that can be used to update the distribution. However, it is not clear how best to combine the rankings from the two sources. At the very least, we have to choose how to weight these two sources of information. There is no reason to believe that the two are equally important or equally likely. While we may have scores associated with these rankings, we cannot blithely interpret these as probabilities. There is much good work on combining evidence from multiple sources. However, the incomplete and unreliable nature of the two sources renders the techniques inapplicable for the interactive data retrieval use case, as we discuss in §VII. Worse yet, the two sources have structural asymmetries as we discuss next.

Implicit feedback. Explicit feedback may be too burdensome to ask of the user. A system cannot usually count on

¹Details of this scoring function are not important for this paper. Our values are obtained from the real system described in [1].

users taking the trouble to correct its mistakes, unless such corrections are on their critical path to completing their task of interest. Therefore, user logs with explicit feedback are likely to be small in size and coverage, if they exist at all. However, observed user behavior after the query can provide implicit feedback:² if a user opts to change the default (top-ranked) option and try another ranked match for a query term we get implicit feedback into an interaction log; but if a user does nothing, it could indicate that she is satisfied (so-called ‘good abandonment’ [2]) or that she is frustrated (bad abandonment).

User-feedback noise. Natural language words may have some genuine ambiguity. Moreover user search behavior may have many oddities: we must admit a range of patterns from hesitant decisions and exploratory choices that may well not be consistent with one another. Given a sufficiently large user feedback log, one can hope that such noise will be suppressed. However, the population of queried terms often form a ‘long tail’ so that many terms have limited feedback and hence are highly susceptible.

DB-encoding noise. The specific DB encoding at both schema and data levels highly influences the system scoring of disambiguation choices. In IMDb, e.g., ‘person’s names are phrased in a more formal manner (e.g., ‘Hanks, Tom’), while names of movie characters are phrased in a more informal manner (e.g., character ‘Tom Hanks’, played in comedy ‘Bamboo Shark’) just like users tend to do in a query. As spelling similarity is one useful measure for matching terms (proper names in particular), the intrinsic scoring is biased against matching ‘tom hanks’ to the person’s name. At the schema level, likewise, IMDb’s table ‘NAME’ (which stores records of people) also obscures the disambiguation of a query like “person hanks”, since the word ‘person’ does not appear in the table or attribute name. Were such encoding bias not present, the DB-intrinsic scoring would be more stable and possibly amenable for characterization as a reliable, non-subjective prior. In view of this bias, the scoring (e.g., as in Example 1), while having indicative value, is not a good probability estimate. Therefore, it is difficult to use the computed scores in a principled manner for composition with another information source.

Research problem. Given a query term comprising one or more words (e.g., ‘tom hanks’), our task is to determine the type of the term. To perform this term type disambiguation, we have available to us two noise-prone information sources. For queries comprising multiple terms, we assume that an upstream process correctly tokenizes the query and then we solve the type disambiguation independently for each token (term).

We devise a novel Bayesian Smoothing solution to this problem, where we use empirically observed noise in the data source to determine its relative weight while accounting for the various expected biases that we discussed above. When user feedback is unambiguously focused on a single answer, as in Example 1, our solution clearly converges to this answer, irrespective of the intrinsic prior, but gives much more credence to the prior when user feedback is more mixed.

List of contributions. In short, the contributions of this

paper are structured as follows.

- **Interactive data retrieval (IDR).** We provide a concrete study of an IDR use case, taking into account real system design and evaluation.
- **Probabilistic model.** We present a bayesian model and an algorithm, Bsmooth, that uses the up-to-date rankings (coming from the two sources) to compensate for the amount of noise found. We also describe how to tune Bsmooth’s parameters given an arbitrary DB-intrinsic scoring profile—so that it can be incorporated by any competing IDR system with its own scoring function.
- **Benchmark-based experiments.** We evaluate the effectiveness of Bsmooth for IDR against an independent and carefully designed benchmark [3]. We also compare it with 9 competitive systems reported in the relational keyword search literature.
- **Empirical findings.** We report results from explicit and implicit user interactions simulated by crowdsourcing for data collection, and then shed light on their comparison.

The paper is organized as follows. In §II we present all the data we have collected for this study. In §III we review entropy as an uncertainty measure. In §IV we present the Bsmooth probabilistic model and algorithm. In §V we evaluate its effectiveness against the benchmark. In §VI we analyze implicit feedback, comparing with explicit feedback. In §VII we present related work, and in §VIII conclude the paper.

II. PREPARATORY DATA COLLECTION

Six related data sets, which we name D0 through D5, are central to describing our work, and to its empirical evaluation. In this section we introduce our notation, these datasets, and our formal problem set up.

Database and Benchmark. D0 is the base dataset, from which everything else is derived. We require that there be a set B of benchmark queries against D0, with known answers. While the specific choice of B and D0 is immaterial for our conceptual development, we will need to specify these for our empirical evaluation. To make matters more concrete for the reader, and to be able to present examples with ease, we reveal our choices of B and D0 here. Note that these specific choices only matter for the evaluation.

For B, we use an independent benchmark developed to evaluate relational keyword search systems in the literature [4], [3]. It follows the standards of the Text Retrieval Conference (TREC) series, and provides relevance assessments on answers to a set of 50 imprecise queries. So it provides B, where $|B| = 50$, and does it separately for each of some popular datasets that have been used by empirical evaluations in related work, e.g., IMDb, Wikipedia, and DBLP.

For D0, we use IMDb release 2009, because of its well-known semantic domain of movies, which fits the scenario of an average user on the web issuing search queries over a relational database whose actual design is unknown to her—recall IMDb’s *TABLE.attribute* design from Example 1. As of Oct 11 (2016), it is available for download as a PostgreSQL dump.³ We have loaded and used it as is. It is the relational database we will disambiguate search queries against.

² Note that when feedback is provided by the user, it is explicit. The more common case is that the user just accepts the default option, and this is implicit feedback. For convenience, we call this entire interaction model as one with implicit feedback.

³<http://www.cs.virginia.edu/~jmc7tp/resources.php>.

In this query on a movie database:
title indiana jones

? Guess what the underlined term means from options:

☐ MOVIE_INFO.info
☐ TITLE.title
☐ NAME.name
☐ CHAR_NAME.name
☐ ROLE_TYPE.role

! Required:

1. Check one or more best matches to the term.
2. Describe what you think each option means.

In this query on a movie database:
title indiana jones

? Guess what the underlined term means from options:

☐ MOVIE_INFO.info
☐ TITLE.title
☐ NAME.name
☒ CHAR_NAME.name
☐ ROLE_TYPE.role

! Required:

1. Choose what you think is "the" best match to the term.

Fig. 1. Crowd task designs to collect explicit (left) and an abstraction of implicit (right) user feedback in a cost-effective simulation of a real system.

The 50 numbered search queries in B, when issued against IMDb, give rise to a total of 62 query terms. The remaining 5 data sets each have scores for each possible type match for each of these 62 query terms. As an aid to memory, we optionally add a character string after each data set number. Thus, we say D1-INTR or D1 to mean the same thing.

D1-INTR. (DB-intrinsic source). We first ran the benchmark queries on our system [1] to get the DB-intrinsic scoring (and induced ranking) for each of the 62 query terms. Example 1 is based on such scoring for query term ‘tom hanks.’ Overall, this dataset D1-INTR has 60% precision-at-rank-one (P@1) accuracy, as we will see in detail later in §V.

Then we used crowdsourcing to simulate explicit and implicit user feedback as a data collection task [5], preceding any modeling carried out by us.⁴ In fact, crowdsourcing has recently been pointed out to be a cost-effective experimentation paradigm to simulate user interactions in Interactive Information Retrieval [6]. We designed two crowd tasks and deployed them at the Microworkers.com platform, paying \$0.10 USD per task instance. The first experiment simulates explicit feedback.

D2-EXPL. (Explicit user feedback). D2 stores the observed counts for the candidate matches as explicitly stated by typical users. For each of the 62 query terms, 10 different workers have been recruited.⁵

In each task instance we ask the worker what the best matches (one or more) are for the term. For example, we refer to query Q22 in the benchmark set B to reproduce here our crowd task design, see Fig. 1 (left). Regarding this design:

- The term is presented in the context of its query to simulate the system scenario more realistically;
- The crowd workers are assumed to be unfamiliar with the IMDb types, i.e., have no knowledge of the database structural details;
- The order of the IMDb types is assigned randomly in each instance of the task (seen by one worker).
- To push workers to process the task cognitively, we require them to describe their choices briefly; their descriptions, being unstructured, are not useful to our results compilation — only the options they check;

⁴In this paper we do not refer to crowdsourcing ‘models’ at all, but only use crowdsourcing for experimental data collection based on best practices [5].

⁵Although a sample size of 10 may look small, it is more realistic as most queried terms may fall in the log’s ‘long tail,’ having limited feedback; besides, we will see later in §V that it has been enough to successfully disambiguate most terms, so it is hard to see how a larger sample could improve our settings.

D3-IMPL. (Implicit user feedback). In D2, we required user feedback for every task. Now in a second crowd experiment (Fig. 1, right), we give users the option to accept the system provided default and provide no feedback. For each of the 62 terms the task is re-instantiated to 10 workers with no participation in the first experiment. This task design has all properties listed above except that two new properties hold:

- A default choice is given. We pick it from the highest-scored option (top-1st) according to dataset D1-INTR, which as mentioned has 60% P@1 and is available to us up-front; e.g., for query term ‘indiana jones’ the default is CHAR_NAME.name, for ‘gone with the wind’ such a default is TITLE.title;
- It is single-choice, while the task design for explicit feedback is multi-choice. This is to reproduce an optional user interaction, where in fact only one option can be chosen at a time.

So this second experiment is a simulation of implicit feedback. D3 stores the observed counts for candidate matches abstracted to be implicitly indicated by users for each of the query terms.

D4-RAND (Random feedback). For each query term, we have sampled from the uniform distribution an integer $0 \leq c \leq 10$ as counts for each disambiguation option. Thereby, we obtain a dataset that reproduces the scenario of a noisy interaction log. D4 will be useful to test how proposed methods behave in the presence of clearly noisy data — a scenario that has to be accounted for in a real system that takes user feedback as input and changes its behavior in response.

D4 stores the observed counts for candidate matches that results from random user input. This really poor result contrasts with D5, described next.

D5-BENCH. (Benchmark’s relevance assessments). D5 is the golden answer key, with the known correct matches for each query term. For each of the 50 imprecise queries in B, the benchmark informs which tuples in the database (D0) are considered relevant — as a binary classification, so each tuple is either relevant or not.⁶

From such tuple-level classification we derive an attribute-level classification as follows. Consider, e.g., query Q7 — which has a single term — and its relevant tuples R7.

Q7. ‘tom hanks’			
R7.	NAME	id	name
		393050	Hanks, Tom

⁶In adherence to the Cranfield paradigm [7], TREC does not distinguish between highly relevant and partially relevant results (cf. [4, p. 6]).

Now recall the 5 options to disambiguate ‘tom hanks’ from Example 1 (rendered by intrinsic D1, by the way). From R7, we know straightforwardly that the one relevant match for the term is NAME.name. We do the same for all of the 62 terms. A highlight is that, when converted from tuple to attribute level, the relevance assessments turn out to be all singletons, i.e., each term has exactly one relevant type to disambiguate it. Query (term) Q14, e.g., has (R14) 3 tuples assessed relevant all of which ‘aggregate’ to TITLE.title.

Q14. ‘lord of the rings’

	TITLE	id	title	production_year
R14.		513253	The Lord of the Rings:...	2003
		513256	The Lord of the Rings:...	2002
		513250	The Lord of the Rings:...	2001

This is how dataset D5 is rendered, gathering the specific IMDb types that are the golden answer key matches to the query terms in the benchmark.

III. ENTROPY AS UNCERTAINTY MEASURE

Since the scores obtained from the intrinsic and extrinsic sources are derived independently, using completely different techniques, an important question to ask is how correct each source is for a particular term. Of course, without knowing the right answer, our system has no way to judge absolute correctness. However, it can judge how certain any source is, using classic entropy measures. Intuitively, we could rely more upon a source that is more certain, in a manner that we will make mathematically precise in the next section.

Given a *scoring vector* $\vec{\theta} = (\theta_1, \theta_2, \dots, \theta_k)$ where $\theta_i \geq 0$ for all $1 \leq i \leq k$ and $\sum_i \theta_i = 1$, that is, $\vec{\theta}$ is a probability distribution, Shannon’s entropy (Eq. 2) gives a measure of the expected surprise (uncertainty) in it [8].

$$H_b(\vec{\theta}) = -\sum_i \theta_i \log_b \theta_i \quad (1)$$

Base b defines a particular unit, which for $b = 2$ is bits. Entropy is a bounded function. For vector $\vec{\theta}$ with $k > 0$ possibilities, it is minimum and maximum (resp.) when $\vec{\theta} = (1, 0, \dots, 0)$ and $\vec{\theta} = (1/k, 1/k, \dots, 1/k)$. Entropy measures how undecided vector $\vec{\theta}$ is. For any k , we have $0 < H_b(\vec{\theta}) \leq \log_b k$. So we can plug Eq. 2 to normalize entropy in the $[0, 1] \subset \mathbb{R}$ interval. Our measure is now unified for any $k \in \mathbb{N}$ and base b for having a comparable threshold.

$$H(\vec{\theta}) = H_b(\vec{\theta}) / \log_b k \quad (2)$$

We can also easily convert counts from an observation vector $\vec{n} = (n_1, \dots, n_k)$, where $n_i \in \mathbb{N}$, to a scoring vector like $\vec{\theta}$ above using relative frequency $\theta_i := n_i / \sum_i n_i$. If we view the DB-intrinsic scoring as a prior hypothesis and the DB-extrinsic counts (whether D2 or D3) as an observation, we can use $H(\vec{\theta})$ to measure how ‘regular’ evidence data is towards the most frequently observed outcome in the sample space. In Example 1 (contd.), which is derived from D3-IMPL, we have 8 counts for NAME.name out of 10 for term ‘tom hanks.’ Its counts vector has $H(\theta) \approx 0.397$, for normalized $0 < H \leq 1$. When the distribution is very skewed, like in this case, we will have low entropy—meaning strong evidence towards the most favored outcome.

We define *ambiguity threshold* to be an entropy value that

TABLE I. Entropy mean \bar{H} and standard deviation (stdev).

Dataset	Entropy mean $\bar{H} \pm$ stdev.
D1-INTR	$98.47 \pm 01.83\%$
D2-EXPL	$48.13 \pm 18.32\%$
D3-IMPL	$19.85 \pm 28.07\%$
D4-RAND	$88.63 \pm 01.34\%$

distinguishes when evidence is “good” from when it is “bad”.⁷ The precise value we choose for this threshold will turn out to be a tuning parameter that we will explore empirically in §V. At this point, we only need the intuition that a data source with entropy less than the threshold provides solid evidence while one with entropy greater than the threshold is more ambiguous about the evidence it provides.

We close this section with some empirical observations regarding the observed entropies in our data sets. Recall that the D1-INTR is already in the form of scoring vectors. For D2-D4 we apply relative frequency to give them the scoring vector form as well. Then, for each of these datasets and each of the 62 terms, we compute $H(\theta)$ by Eq. 2. Table I shows the entropy mean \bar{H} and stdev that we have found. Some observations about Table I are:

- Every IDR system will have its own intrinsic scoring function, which as we have discussed is prone to DB-encoding noise. Our particular D1-INTR has very high entropy mean with very low stdev, which indicates that entropy is nearly the same over queried terms and may not help to indicate (in)accuracy for D1-INTR;
- D2-EXPL has higher entropy mean than D3-IMPL, while for stdev the inverse holds. This can be easily explained by the workers being able to choose multiple choices per task in Fig. 1 (left), but only one choice per task in Fig. 1 (right). In fact D2-D3 have been collected for different purposes: (D2) a closer to “true” disambiguation distribution as a baseline, and (D3) a more realistic interaction log;
- For both D2-EXPL and D3-IMPL, the entropy mean, even with added stdev, is far below D4-RAND’s entropy mean, which means that the feedback provided by a crowd of users can indeed be informative.

The first observation above suggests that the intrinsic source may not be sensitive to entropy, while the two other observations suggest that the extrinsic source is. We will take that into account in the analytical study that comes next, and revisit this assumption later in our evaluation in §V.

IV. PROBABILISTIC MODEL

In this section we define relevant concepts and state the inference problem in light of the general bayesian smoothing model. We then consider progressively more involved solutions, ultimately arriving at our proposed algorithm, Bsmooth.

Def. 1: Let $k \in \mathbb{N}^*$. We say that \vec{x} is a **scoring vector** of size k , and write $\vec{x}(x_1, \dots, x_k)$, if we have $x_i \geq 0$ for all bins $i \in \mathbb{N}^*$ where $i \leq k$ and $\sum_i x_i = 1$.

Def. 2: Let \vec{x} and \vec{y} be two scoring vectors. We say that $\vec{z} = (z_1, \dots, z_k)$ is a **convex combination** of \vec{x} and \vec{y} , and write

⁷Evidence being “bad” does not mean that its probability distribution is wrong, as some terms may be genuinely ambiguous; e.g., ‘terminator’ may well refer to either character name or movie title. However, it may still mean that we should be less likely to trust high-entropy counts vectors as opposed to low-entropy ones. This intuition will be empirically evaluated in §V-B.

$\vec{z} = \vec{x} \oplus \vec{y}$, if there exist $\omega_1, \omega_2 \geq 0$ with $\omega_1 + \omega_2 = 1$ such that $z_i = \omega_1 x_i + \omega_2 y_i$ for all bins $i \in \mathbb{N}^*$, $i \leq k$.

Bayesian inference allows us to arrive at a posterior combining information from two sources — if we take the DB-intrinsic scoring as a prior, which is updated with the DB-extrinsic observations to get a posterior. As the prior and posterior scoring vectors take values in the continuous parameter space, bayesian inference takes the form of *bayesian smoothing* [8]. Since for each instance of disambiguation problem we have (with no loss of generality) a number of $k > 2$ types as options to disambiguate, the model will take the form of a *Dirichlet-multinomial* [8, p. 78].

Remark 1: The extent to which we can consider the DB-extrinsic counts as ‘observations’ in our interactive data retrieval problem (i.e., as empirical counts, even if subject to some smoothing) is still not quite clear. We will proceed from here as if so, to gain depth into the abstract problem, and then revisit this assumption later in §V-B. \square

A. The Bayesian Smoothing Problem

Bayesian smoothing in terms of the dirichlet-multinomial can be obtained from the *dirichlet posterior mean* as given by Eq. 3, where $p(\theta_i | \vec{n})$ is the posterior probability of type $i \in \mathbb{N}^*$, $i \leq k$ as a disambiguation option; and n_i , α_i are components of the observed counts vector \vec{n} and hyper-parameter vector $\vec{\alpha}$ with n , α as the total observations and the hyper-prior concentration parameter.

$$p(\theta_i | \vec{n}) = \frac{n_i + \alpha_i}{n + \alpha} \quad (3)$$

Here, \vec{n} maps naturally to our observation counts on each type i , while $\vec{\alpha}$ maps to our DB-intrinsic scoring vector as a (hyper-)prior. The formula simply gives their weighted sum based on the scale of $\alpha \in \mathbb{R}^+$ against $n \in \mathbb{N}$. The open problem is how to set this relative weight, i.e., the strength of the prior, or alternatively, the weight of evidence. What we really have here is a convex combination problem where the scaling of α must be done at query time.

Problem 1: Let $\vec{x}(x_1, \dots, x_k)$ be a scoring vector rendered by the DB-intrinsic source, and $\vec{n}(n_1, \dots, n_k)$ with $\sum_i n_i = n$ be the counts vector supplied by the DB-extrinsic source, both at query time. For notation and analytical convenience, the counts vectors \vec{n} is normalized into a scoring vector \vec{y} .

So we have two scoring vectors \vec{x} and \vec{y} from (resp.) the intrinsic and extrinsic sources and want to find $\vec{z} = \vec{x} \oplus \vec{y}$ that gives the best results for all queried terms. That is, we seek $\omega_1, \omega_2 \geq 0$ with $\omega_1 + \omega_2 = 1$ such that $z_i = \omega_1 x_i + \omega_2 y_i$ for all $i \leq k$, $i \in \mathbb{N}^*$, such that the correctness of \vec{z} (which we will measure as P@1), averaged over queried terms, is maximum. We call ω_2 the **weight of evidence**. \square

We can just set $\alpha := n(\omega_1/\omega_2)$ to scale $\vec{\alpha}$ accordingly and get the posterior scoring vector $\vec{\theta} \in \mathbb{R}^k$ estimated by bayesian smoothing within the k -simplex continuous parameter space. This is bayesian smoothing where the (hyper)prior has an empirical basis.

B. Baseline Schemes: The Limiting Cases

A first and very simple baseline method to consider is Maximum Likelihood Estimation (MLE), which corresponds to bayesian smoothing with $\omega_2 = 1$ in Problem 1.

Remark 2: Simply set $\omega_2 = 1$ ($\omega_1 = 0$). For reference we call this method **MLE**. \square

MLE considers only the observations, and ignores any prior obtained from an intrinsic model. It can work very well if there are enough observations and they are low entropy. However, such an assumption is too strong. MLE seems too naive and we do not expect it to perform well in general.

The opposite limiting case is to set $\omega_2 = 0$ ($\omega_1 = 1$); that is, to ignore all observations and just stay with the intrinsic model irrespective of the observed implicit user feedback. This is trivial to do, and obviously misses the whole point of improving disambiguation by learning from user behavior.

A smarter technique would choose between these two limiting cases based on which one is more likely to be correct.

Remark 3: Given an ambiguity threshold $D \in [0, 1]$, the weight $\omega_2 \in [0, 1]$ could be defined simply as $\omega_2 = 1$ ($\omega_1 = 0$) if $H(\vec{y}) \leq D$, and $\omega_2 = 0$ ($\omega_1 = 1$) otherwise. This would be a (discontinuous) staircase function we call **STEP**. \square

C. Balanced Convex Combinations

In this subsection, we study balance properties of convex combinations, so as to be able to design less extreme weighting functions.

Example 2: Given two scoring vectors $\vec{x} = (1, 0, 0, 0, 0)$ and $\vec{y} = (1/3, 1/3, 0, 1/3, 0)$, one can ask, say, is $\vec{z} = (1/2, 1/4, 0, 1/4, 0)$ a convex combination of \vec{x} and \vec{y} ? Yes, we can pick $\omega_1 = 1/4$ and $\omega_2 = 3/4$ so that \vec{z} can be written $z_i = (1/4)x_i + (3/4)y_i$ for all bins $1 \leq i \leq 5$. \square

Lemma 1: Let \vec{x} and \vec{y} be two scoring vectors and $t \neq u$ two of their bins. Then a convex combination $\vec{z} = \vec{x} \oplus \vec{y}$ with $z_t = z_u$ exists and is unique if and only if one of these mutually exclusive conditions hold:

- (i) $(x_t - x_u)/(y_t - y_u) < 0$,
- (ii) $(x_t - x_u) = 0$ but $(y_t - y_u) \neq 0$,
- (iii) $(y_t - y_u) = 0$ but $(x_t - x_u) \neq 0$.

Proof: We show the statement by construction of an abstract convex combination \vec{z} satisfying $z_t = z_u$. The conditions stated by the lemma are then tied up to the existence of a unique pair $\omega_1, \omega_2 \geq 0$ as the solution set of a linear system in these two variables. See Appendix IX-A. \blacksquare

Note that of the three cases in the lemma, only the first is a “true” convex combination: the other two are really corner cases where we ignore one of the two inputs.

Example 2: (contd.) For the vectors $\vec{x} = (1, 0, 0, 0, 0)$ and $\vec{y} = (1/3, 1/3, 0, 1/3, 0)$, suppose that we wanted to find any convex combination $\vec{z} = \vec{x} \oplus \vec{y}$ with $z_2 = z_4$. The one that has been presented satisfies it — recall $\vec{z} = (1/2, 1/4, 0, 1/4, 0)$ rendered out of $\omega_1 = 1/4$, $\omega_2 = 3/4$. Yet there are infinitely many pairs $\omega_1, \omega_2 \geq 0$ with $\omega_1 + \omega_2 = 1$ that we could have picked so that $z_2 = z_4$. In fact, by Lemma 1 we should not expect uniqueness here as we have both $(x_2 - x_4) = (y_2 - y_4) = 0$. In contrast, if we want $z_1 = z_3$, that is not possible because $x_1 > x_3$ and $y_1 > y_3$. \square

Def. 3: Let $\vec{x} = (x_1, \dots, x_k)$ be a scoring vector and $1 \leq t \leq k$ one of its bins. We say that t is a **top bin** of \vec{x} , and write $t \in [\vec{x}]$, if $x_t \geq x_i$ holds for all bins $1 \leq i \leq k$.

Def. 4: Let $\vec{x} = (x_1, \dots, x_k)$ and $\vec{y} = (y_1, \dots, y_k)$ be two scoring vectors with top bins $t \in [\vec{x}]$ and $u \in [\vec{y}]$. We say

TABLE II. Two separate rows: entry \vec{x}_i in first row applies to each entry \vec{y}_j in second row, rendering one curve in Fig. 2. That is, first row shows the prior skew levels from \vec{x}_0 to \vec{x}_6 . Second row shows the several points (entropy levels) \vec{y}_0 to \vec{y}_{20} .

PRIOR	$\vec{x}_0(.499, .501)$	$\vec{x}_1(.4275, .5725)$	$\vec{x}_2(.4, .6)$	$\vec{x}_3(.3, .7)$	$\vec{x}_4(.2, .8)$	$\vec{x}_5(.1, .9)$	$\vec{x}_6(.001, .999)$
POINT	$\vec{y}_0(.999, .001)$	$\vec{y}_1(.975, .025)$	$\vec{y}_2(.950, .050)$...	$\vec{y}_{18}(.550, .450)$	$\vec{y}_{19}(.525, .475)$	$\vec{y}_{20}(.501, .499)$

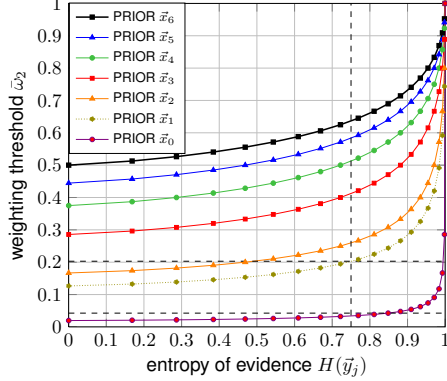


Fig. 2. Thresholds to balance out disagreed top bins for a class of priors of fixed skewness. Legend keys follow the order the curves appear in the vertical axis. The more skewed the prior the higher lies its curve. For $\vec{y}(.7875, .2125)$, e.g., which has $H(\vec{y}) \approx .75$, one needs only $\omega_2 \approx .0035$ to balance \vec{x}_0 .

that t and u are **disagreed top bins** (of \vec{x} and \vec{y}), and write $t \asymp u$, if either $t \notin \lceil \vec{y} \rceil$ or $u \notin \lceil \vec{x} \rceil$ or both.

Theorem 1: Let $\vec{x} = (x_1, \dots, x_k)$ and $\vec{y} = (y_1, \dots, y_k)$ be two scoring vectors with top bins $t \in \lceil \vec{x} \rceil$ and $u \in \lceil \vec{y} \rceil$. If these are disagreed top bins $t \asymp u$, then a convex combination $\vec{z} = \vec{x} \oplus \vec{y}$ with both $z_t = z_u$ exists and is unique.

Proof: From Lemma 1 and Def. 3–4, see Appendix IX-B. ■

Note that when the intrinsic and extrinsic vectors \vec{x} and \vec{y} agree on their top bins, there is little for us to do. The interesting case is when they have disagreed top bins, and this is precisely the case covered by Theorem 1. The theorem states that we can solve a linear system in two variables, $\omega_1, \omega_2 \geq 0$ to find a balanced combination where two top choices are equally highly scored. Intuitively, this balance point is important because the final top choice answer tends to be decided one way or the other whether we increase ω_1 or ω_2 .

To develop a better intuition for this balance point, we consider a wide range of values for \vec{x} and \vec{y} as shown in Table II. The goal of using such a range of values is to explore balance conditions in terms of the varying entropy measures $H(\vec{y})$, as given by Eq. 2. To avoid degenerate cases, we use $(.499, .501)$ rather than $(.5, .5)$ and $(.001, .999)$ rather than $(0, 1)$. We consider every entry in the first row paired with every entry in the second row. For each pair \vec{x}_i, \vec{y}_j , where \vec{y}_j has its own entropy measure, we solve the linear system in variables $\omega_1, \omega_2 \geq 0$. Fig. 2 shows the results.

The higher is the entropy, the higher is the required threshold ω_2 for balance, except for the least skewed \vec{x}_0 (violet) whose ω_2 is nearly constant most of the domain.

D. Baseline Scheme: Linear Weighting

Example 3: Let $\vec{x} = (.45, .43, .12, 0, 0)$ and $\vec{y} = (.08, .45, .47, 0, 0)$ be two scoring vectors. Note that we have different sets of top bins $\lceil \vec{x} \rceil = \{1\}$ and $\lceil \vec{y} \rceil = \{3\}$. Then by Theorem 1 we know that a unique convex combination $\vec{z} = \vec{x} \oplus \vec{y}$ exists with $z_1 = z_3$. In fact, this is $z_i \approx .54x_i + .46y_i$ so that $\vec{z} \approx (.28, .44, .28, 0, 0)$. Now we have $\lceil \vec{z} \rceil = \{2\}$, which is

not one of the top bins of \vec{x} nor of \vec{y} .⁸

Entropy is $H(\vec{y}) \approx .569$, which is not very high. So if the ambiguity threshold is, say, $D = .75$, then we have $\text{reg}_D(\vec{y}) = 1$. Now recall the baseline method of STEP from Remark 3. The top bin picked would be $\lceil \vec{z} \rceil = \{3\}$, just as if out of MLE from \vec{y} . □

We see that STEP has an undesirable discontinuity jump: a very small change in the extrinsic scoring’s entropy $H(\vec{y})$, say, a single user clicking one thing differently, say in the process of exploring her disambiguation options, can completely change the system output. Instead, we would like that such small changes should not add too large a change in its weight ω_2 . At the very least, we would like $\omega_2 : [0, 1] \rightarrow \mathbb{R}$ to be a continuously differentiable function: its first derivative exists all over $[0, 1]$ and is itself a continuous function.

Remark 4: Given the entropy $H(\vec{y})$ of extrinsic scoring \vec{y} , its weight $\omega_2 : [0, 1] \rightarrow [0, 1]$ could be defined as a linear function $\omega_2(H) = \beta_0 - \beta H$ for some constants $\beta_0, \beta > 0$. Note the negative sign (with $\beta > 0$) to meet the intuition of inverse correlation: the higher is $H(\vec{y})$, the less likely is \vec{y} to be accurate. Now if we also take boundary conditions $\omega_2(0) = 1$ and $\omega_2(1) = 0$ as two related constraints, then we easily infer $\beta_0 = \beta = 1$ and get more specific $\omega_2(H) = 1 - H$. We call this method **LINEAR**. □

E. Proposed Scheme: Log-linear Weighting

By Example 4 we illustrate why a linear weighting profile cannot be adequate in general.

Example 4: Let us revisit Example 3 with $\vec{x} = (.45, .43, .12, 0, 0)$, but now with $\vec{y}' = (.08, .35, .37, .1, .1)$ instead. By Lemma 1 the unique convex combination that balances out top bins $\lceil \vec{x} \rceil = \{1\}$ and $\lceil \vec{y}' \rceil = \{3\}$ exists and is precisely $z_i \approx .468x_i + .532y'_i$ so that $\vec{z} \approx (.253, .387, .253, .053, .053)$. Again we have $\lceil \vec{z} \rceil = \{2\}$, not a top bin of \vec{x} nor of \vec{y}' .

But here we have entropy $H(\vec{y}') \approx .885$, which is fairly high. The same ambiguity threshold, $D = .75$, would make $\text{reg}_D(\vec{y}') = 0$. Thus we have: by MLE, $\lceil \vec{z} \rceil = \{3\}$; by STEP, $\lceil \vec{z} \rceil = \{1\}$; and by LINEAR, $\omega_1 \approx .885, \omega_2 \approx .115$, then $\vec{z} \approx (.407, .421, .149, .012, .012)$ and therefore $\lceil \vec{z} \rceil = \{2\}$. □

It is not surprising that LINEAR fails to bend to penalize entropy properly in the neighborhood of the ambiguity threshold $H = D \pm \delta$, since it has a constant rate of change w.r.t. entropy, $\omega'_2(H) = -1$. Furthermore, for any chosen ambiguity threshold, D , we would like the weight to be set as $\omega_2(D) = \bar{\omega}_2$ plotted in Fig. 2 for each prior. Recall the reason for this crucial constraint from §IV-C. At the ambiguity turning point D we do *not* know which scoring to favor, the extrinsic or the intrinsic. LINEAR — being a straight line — obviously cannot fit all constraints in $P = \{(0, 1), (1, 0), (D, \bar{\omega}_2)\}$ with third point $(D, \bar{\omega}_2)$ virtually anywhere in the plane $[0, 1]^2$.

⁸This is really not a problem as this “consensus” top bin $\lceil \vec{z} \rceil = \{2\}$ is brought forth naturally out of balancing the two vectors, and not by an inferential scheme that pushes it through individually at the bin granularity level — cf. discussion in VII of Dempster-Shafer’s theory applied in related work.

To address this shortcoming, we desire a function that can fit the extreme points $(0, 1)$, $(1, 0)$ but still have room to fit the third point $(D, \bar{\omega}_2)$. Furthermore, consider that our extrinsic scoring on a given term is either very skewed or very entropic, (resp.) near complete certainty or complete uncertainty. At such states, we want to change the weight of evidence only slowly, with some lag, say, to respond to one or another user with exploratory behavior. We want to implement inverse correlation (cf. Remark 4) linearly otherwise. This is achieved with the *logit transform* [9].

Remark 5: The logit transform is given by $\text{logit}(\theta) = \ln[\theta/(1 - \theta)]$, and is also called log-odds. We model the relationship between ω_2 and H to be log-odds linear, as given by Eq. 4 for constants $\beta_0, \beta > 0$.

$$\ln \frac{\omega_2}{1 - \omega_2} = \beta_0 - \beta H \quad (4)$$

The logit is the inverse of the logistic function: let both sides be exponents of base e and easily solve for ω_2 to get Eq. 5.

$$\omega_2(H) = \frac{1}{1 + e^{-(\beta_0 - \beta H)}} \quad (5)$$

We call this method **LOGIT**. \square

We assess LOGIT against any linear function in Fact 1.

Fact 1: Let $P = \{(0, 1), (1, 0), (D, \bar{\omega}_2)\}$ be a set of points in the plane $[0, 1]^2$, where $\bar{\omega}_2$ is taken over each of the weighting thresholds considered — acquired out of Theorem 1 given the DB-intrinsic profiles shown in Table II. To instantiate this fact, we will keep the same $D = .75$ we have been using in previous examples, and revisit this parameter in §V-B.

Now let the lack of fit for a function $f: [0, 1] \rightarrow \mathbb{R}$ be defined by least squares $L_f = \sum_{(a,b) \in P} (b - f(a))^2$; and $f_1(x) = \beta_0 - \beta x$ stand for linear functions (e.g., LINEAR) whereas $f_2(x) = 1/(1 + e^{-(\beta_0 - \beta x)})$ stand for LOGIT over $\beta_0, \beta > 0$.

If $\beta_0, \beta > 0$ are tuned optimal to minimize L_{f_1}, L_{f_2} independently based on an off-the-shelf curve fitting solver, then we have $L_{f_2} \ll L_{f_1}$. (the optimal LOGIT beats the optimal linear function in all seen cases by orders of magnitude).

Proof: This fact is validated in Appendix § IX-C, where we present Tables III–IV with the optimal $\beta_0, \beta > 0$ for f_1, f_2 over each $\bar{\omega}_2$ and the corresponding L_{f_1}, L_{f_2} . \blacksquare

Fact 1 has been verified with several other values $0 < D < 1$, and is decisive about which weighting profile to choose. See Fig. 3 for a visual reference, and note that it shows all the weighting profiles we have seen so far.

We observe that LINEAR, in particular, could only have a small L_f given some $(D, \bar{\omega}_2)$ if this point happens to be close enough to the line $\omega_2(H) = 1 - H$. For this reason, entry \bar{x}_2 for $(.75, .260)$ in Table III has the smallest L_f . Even if we seek another linear profile with $\beta_0 \neq \beta \neq 1$, we would have to deviate from the other constraints $(0, 1), (1, 0)$. LOGIT, on the other hand, is also up to 2 parameters and, most importantly, it can fit the third point anywhere in the plane with smaller deviation from $(0, 1), (1, 0)$. LOGIT can have even a nearly linear shape if so desired.

Even after going through the specified arbitrary point, LOGIT still has one remaining degree of freedom, which determines how fast it changes. At one extreme, it could be almost linear. At another extreme, it could be almost a step function. Ideally, we want it somewhere in between. We manage this by

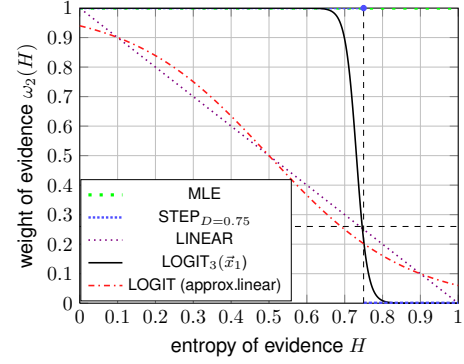


Fig. 3. Various weighting profiles from MLE and STEP to LINEAR. Profile $\text{LOGIT}_3(\bar{x}_1)$, with $|P| = 3$, is fit to prior \bar{x}_1 (Table IV) with $\beta_0 = 50.270$, $\beta = 68.766$. LOGIT profile (approx.linear) is arbitrary with, say, $\beta_0 = 2.75$, $\beta = 5.5$, to show LOGIT's flexibility.

setting the slope of the function near the ambiguity threshold. Intuitively, we are choosing the *slope* near the ambiguity threshold in addition to the *intercept information* $(D, \bar{\omega}_2)$. For this purpose, we define a fourth constraint $(D + \delta, \bar{\omega}'_2)$, where $\delta \in \mathbb{R}$ is small (e.g., $\delta = .05$) and defines a neighborhood of D^+ . We let $P' := P \cup \{(D + \delta, \bar{\omega}'_2)\}$. This value $\bar{\omega}'_2$ is taken from the inverse slope $-m^{-1}$, where slope m is positive and computed as usual $m = |\bar{\omega}_2(D + \delta) - \bar{\omega}_2(D)| / (D + \delta - D)$. The intuition, as just mentioned, is to enforce the opposite behavior w.r.t. Fig. 2: the more vertical the slope is, the smoother the weighting profile can be. Fact 1 is invariant to the extension of P to P' .

The above reasoning is encoded into Algorithm 1. In Appendix IX-D we give final tuned parameters for LOGIT_4 , with $|P| = 4$ where $P' = \{(0, 1), (1, 0), (.75, \bar{\omega}_2), (.80, \bar{\omega}'_2)\}$. The ambiguity threshold $D = .75$ is discussed later in §V.

Remark 6: We propose **LOGIT** to be built into **Bsmooth** (see next). Overall, we have met inverse correlation (cf. Remark 4) and been able to extract information from the DB-intrinsic scoring in spite of its limited informativeness (cf. Table I in §III). We will revisit these intuitions in §V-B. \square

F. The Bayesian Smoothing Algorithm

The complete pipeline is encoded into Algorithms 1-2. A high level description of our approach is as follows.

- 1) Weight of evidence ω_2 is really a predictive probability,

Algorithm 1 Prior's profiling.

1: **procedure** PRIORSPROFILING(\vec{x} : scoring vector; D : ambiguity thresh.; δ : neighbd.; \mathcal{S} : population of terms)

Part I: weighting threshold intercept

2: find entropy mean $\bar{H}(\vec{x})$ over \mathcal{S} \triangleright applies Eq. 2
3: find $\bar{\theta}_{prior}(x_1, x_2)$ such that $H(\theta_{prior}) \approx \bar{H}(\vec{x})$
4: find $\bar{\theta}_{evid}(y_1, y_2)$ such that $H(\theta_{evid}) \approx D$
5: $\bar{\omega}_2 \leftarrow \lambda / (1 + \lambda)$, where $\lambda = |x_1 - x_2| / |y_1 - y_2|$
6: $P \leftarrow \{(0, 1), (1, 0), (D, \bar{\omega}_2)\}$ \triangleright initializes P

Part II: weighting threshold slope

7: $m \leftarrow |\bar{\omega}_2(D + \delta) - \bar{\omega}_2(D)| / \delta$ $\triangleright \delta = (\mathcal{J} + \delta) - \mathcal{J}$
8: $\bar{\omega}'_2 \leftarrow \bar{\omega}_2(D) - m^{-1} \delta$ \triangleright intercept $- \text{slope}^{-1} \cdot \delta$
9: $P' \leftarrow P \cup \{(D + \delta, \bar{\omega}'_2)\}$
10: find $\beta_0, \beta > 0$ such that LOGIT's lack of fit to P' is min.
11: **return** β_0, β

Algorithm 2 Bayesian smoothing.

```

1: procedure BSMOOTH( $\vec{x}, \vec{n}$ : vectors;  $\beta_0, \beta$ :  $\mathbb{R}_{>0}$ )
Ensure: returns a convex combination  $\vec{\theta} = \vec{x} \oplus \vec{y}$ 
2:    $\vec{y} \leftarrow \vec{n}$  scaled to 1
3:    $H \leftarrow (\sum_i^k -y_i \log y_i) / \log k$  ▷ Shannon's Eq. 2
4:    $\omega_2 \leftarrow 1 / (1 + \exp(-\beta_0 + \beta H))$  ▷ LOGIT (Eq. 4)
5:    $\omega_1 \leftarrow 1 - \omega_2$ 
6:    $\alpha \leftarrow n(\omega_1/\omega_2)$  ▷ sets the scale of  $\alpha$ 
7:    $\vec{\alpha} \leftarrow \vec{x}$  scaled to  $\alpha$ 
8:   for  $i \leftarrow 1$  to  $k$  do ▷ Dirichlet-multinomial (Eq. 3)
9:      $\theta_i \leftarrow (n_i + \alpha_i) / (n + \alpha)$ 
10:   $\vec{\theta} \leftarrow \vec{\theta}$  scaled to 1
11:  return  $\vec{\theta}$ 

```

$p(\vec{y})$, about whether the extrinsic source \vec{y} is trustable. To do bayesian smoothing wisely, we need to find $\omega_2(H)$, i.e., to find $p(\vec{y} | H(\vec{y}))$, which should have a LOGIT profile.

- 2) Besides the two obvious points $\{(0, 1), (1, 0)\}$ in the plane $[0, 1]^2$, we need to fit one or two more constraints to define the behavior of our weighting scheme at the ambiguity threshold, H_D , and nearby, $H_D + \delta$.
- 3) Intuitively, since H_D is the turning point of regularity for (in-)accuracy, at H_D the weighting scheme should combine our two sources towards balance. Theorem 1 gives theoretical support by defining conditions for balance.
- 4) Now given one's own specific prior profile we can find constraints $P' = \{(0, 1), (1, 0), (H_D, \bar{\omega}_2), (H_D + \delta, \bar{\omega}_2')\}$. This is done (Algorithm 1), only once, offline. The fitted logistic curve is just $p(\vec{y} | H(\vec{y}))$, which we were seeking. It provides parameters $\beta_0, \beta > 0$ into Algorithm 2.
- 5) Given a new term disambiguation problem instance, by Algorithm 2 we simply compute $H(\vec{y})$ and get $p(\vec{y} | H(\vec{y}))$, namely, ω_2 , and then apply bayesian smoothing given ω_2 .

Clearly, Bsmooth is bounded in time $O(k)$, where k is the number types retrieved, fixed in query and data complexity. Considering IMDb and other benchmarked databases [4], a typical value is $k = 5$, and yet for a very complex DB schema one could think of $k \leq 5c$ for some small $c \in \mathbb{N}^*$.

V. EFFECTIVENESS EVALUATION

We use the benchmark dataset D5-BENCH (cf. §II), developed independently [4], [3], as a gold standard to evaluate accuracy rates in several settings.

A. Evaluation Metrics

The accuracy metric that is central to this paper is P@1 [10]. This is because the real system that we are modeling must avoid mistakes on the default option that is presented to users as disambiguation to a query term. Note that every accuracy mistake burdens the user with an interaction that is then necessary to satisfy her information need. ‘Recall,’ by comparison, is not important because $k \in \mathbb{N}$, the number of candidate schema elements that are potentially relevant, is bound to be small. To make sure P@1 is not too rigorous, we will also cover (the more flexible) ‘mean reciprocal rank.’

Precision-at-rank-1 (P@1). For each term P@1 scores 1 when the highest-ranked result is relevant, and 0 otherwise. For the ranking shown for query term ‘tom hanks’ in Example 1, we have P@1 zero. We will aggregate P@1 by arithmetic mean over the 62 terms.

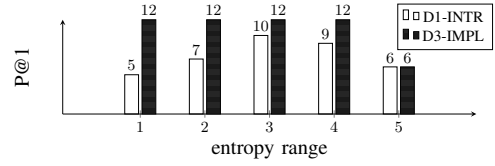


Fig. 4. Histograms of P@1 hits over ranges of increasing entropy for datasets D1-INTR and D3-IMPL.

Mean Reciprocal Rank (MRR). This is the reciprocal of the highest-ranked relevant result for each query term averaged over all of them (see Eq. 6). In the ranking shown for ‘tom hanks’ in the first part of Example 1, we have $RR=1/2$ because the one relevant result NAME.name is ranked at position 2.

$$MRR = \frac{1}{n} \sum_{i=1}^n \frac{1}{rank(i)} \quad (6)$$

B. Study of Accuracy Profiles

We have designed Bsmooth under assumption that entropy is informative for the extrinsic source (evidence) but not for the intrinsic one (hypothesis). Now it is time to revisit and see if it carries on for interactive data retrieval.

Fig. 4 shows P@1 scores for D1-INTR and D3-IMPL across ranges of increasing entropy. Given the entropy of the scoring vector for each of the 62 terms, we formed 5 entropy ranges of sizes 12, 12, 12, 13, 13, covering D1 and D3 from the lowest- to the highest-entropy scored terms. We observe:

- For D1-INTR (37/62 hits in P@1), no (inverse) correlation is seen between entropy and accuracy. The intrinsic model is subject to noise in unpredictable ways.
- For D3-IMPL (54/62 hits in P@1), we do observe inverse correlation, which does not seem to be linear (6 mishits in the last range and 1 in the range before). Note carefully that it is only from some entropy threshold on that we observe a significant drop in accuracy.

We see that implicit user feedback (D3) have high P@1 accuracy. So the user interactions really qualify for ‘observed’ counts for bayesian inference — this is a non-obvious finding for interactive data retrieval.

C. Internal Evaluation

The histogram for D3-IMPL in Fig. 4 provides empirical support to set the ambiguity threshold $H_D \approx .75$. The results we present here are based on this tuning.⁹

Fig. 5 shows results of the internal evaluation. We concentrate the discussion on P@1 (left), as MRR (right) is shown only to make sure the P@1 results preserve their structure in a more flexible metric — and they do.

The main observation is that LOGIT, the proposed bayesian smoothing strategy, can roughly retain the accuracy of INTR even under D4-RAND data, while significantly beat it under both D2-EXPL and D3-IMPL data. Its P@1 scores are also better than the baseline strategies MLE, STEP and LINEAR in the most important datasets D4-RAND and D3-IMPL, which simulate user feedback with skepticism and avoid user burden.

⁹To review and fine tune their own ambiguity threshold, users should simply plot an histogram like Fig.’s 4 for the variant of D3-IMPL obtained from their own interaction log, either real or simulated, given the golden answer keys from their own variant of D5-BENCH.

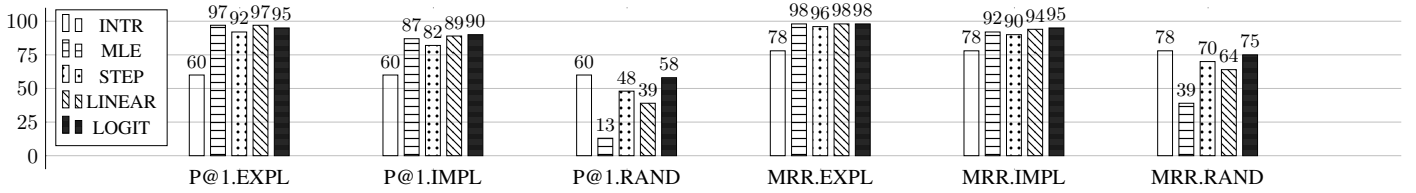


Fig. 5. P@1 and MRR in % over the $n=62$ query terms with number of retrieved types $k=5$. The metrics are shown for baseline choices INTR, MLE, STEP, LINEAR and proposed technique LOGIT over datasets D2-EXPL, D3-IMPL, D4-RAND.

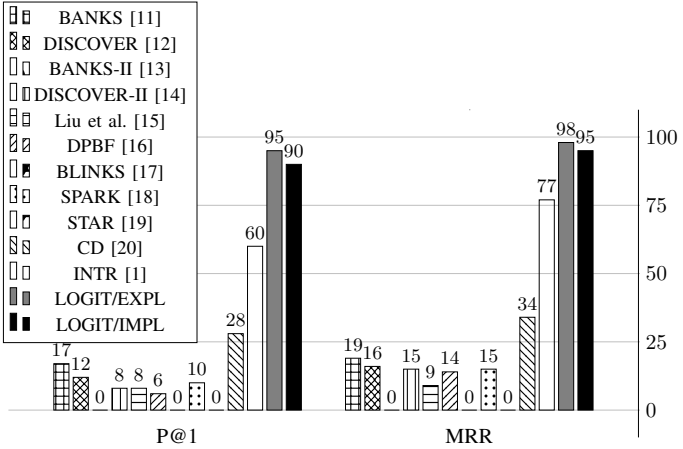


Fig. 6. P@1 and MRR over 50 queries with 100 retrieved tuples each. The metrics are shown for related systems in the literature (source: [3]) and LOGIT/EXPL and /IMPL.

The differences in P@1, of course, are not large because we have deliberately moved from one strategy to the other in seek of every small improvement.

STEP lacks a continuous decay profile at the neighborhood of the ambiguity threshold, and LINEAR fails to penalize high entropy fast enough. MLE is in fact not solid to be deployed into a system strategy. We see that under RAND data, say when a term is very ambiguous or users behave undecidedly as a crowd, it would make the system vulnerable. INTR is more often accurate than not (60%), yet not highly accurate. It can be significantly improved by learning from implicit feedback as an extrinsic source. These are robust results towards LOGIT-based bayesian smoothing.

D. External Evaluation

We have also obtained from the benchmark’s authors the accuracy rates of 10 competitive systems in the literature on the exact same IMDb dataset (D0) and 50 queries [3]. We do not have, however, the specific tuples returned in order to ‘aggregate’ them into types. The ‘external’ evaluation comparison is then indirect in the following sense.

To project our accuracy rates onto tuple level, we assume an accuracy hit in P@1 only if we disambiguate accurately all the terms in a query. Otherwise we take P@1 zero and RR to be the least over all terms in that query. Fig. 6 shows the accuracy rates for the external evaluation. Not surprisingly, the addition of a learning layer (LOGIT/D3-IMPL) on top of a DB-intrinsic source (INTR) strongly outperforms the literature systems [11], [12], [13], [14], [15], [16], [17], [18], [19], [20].

When a system has score zero, it is because according to the benchmark authors (not us) it takes a really large time to respond [3]. Unpredictable response times are also reported by

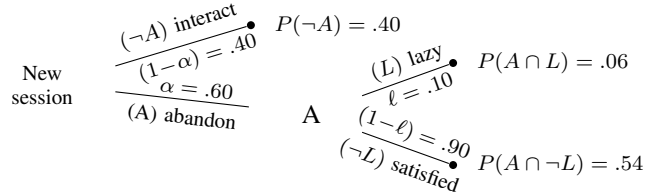


Fig. 7. Probability tree diagram and a valuation example. The abandonment rate (α) and ‘good’ abandonment rate $\alpha(1-\ell)$ are free parameters.

Baid et al. for many of these systems [21]. Although we are not concerned with performance evaluation here, it is a fair point that unacceptable response time is prohibitive for effectiveness [3]. Excessive response time is pointed out as related with the generation of large data graphs [21].

An in-depth evaluation of those systems is presented in [3]. Here we are concerned with evaluating a learning layer that is fairly independent of any particular method to produce the DB-intrinsic scoring, and takes only $O(k)$ time (cf. §IV-F).

VI. EMPIRICAL ANALYSIS OF IMPLICIT FEEDBACK

Recall that explicit user feedback is desirable, but may be unavailable or limited. As such, we expect to be limited to implicit user feedback in most practical situations. We saw above that Bsmooth’s accuracy with D3-IMPL was only slightly worse than its accuracy with D2-EXPL. In this section, we look at this issue more carefully.

The central problem with implicit feedback is to interpret correctly the case where a worker just accepts the default choice, as if in session abandonment. How could we know whether this happened due to user satisfaction with the default or user indifference (or ‘laziness’)? Fig. 7 shows the corresponding estimation scheme as a probability tree. There are two parameters: $\alpha := P(A)$ and $\ell := P(L)$, where A is the event of abandonment and L the event of ‘laziness.’

We can observe the rate of session abandonment, α , but not the ‘laziness’ parameter ℓ . If we have some explicit feedback available, that can be used as a baseline to estimate ℓ . We need to see if there is a significant ‘laziness’ effect behind implicit feedback. So for each term, e.g., ‘angelina jolie’, we take the default option (NAME.name) shown to the worker for the generation of D3-IMPL, and compared the feedback rate (with possible laziness effect) of that particular option against its corresponding rate from D2-EXPL (baseline).

Fig. 8 (top) shows the results, term-wise, for a sample of 10 terms. For query term ‘angelina jolie,’ we see a D2-score of 71%, and a D3-score of 80%. Thus for this particular term we have an estimated laziness of $\ell = 9\%$. Across the other 9 terms, we see, of course, some variance on ℓ . Over all the 62 terms, we have mean $\ell = 5.1 \pm 13\%$.

Now, to make sure the laziness effect carries on even

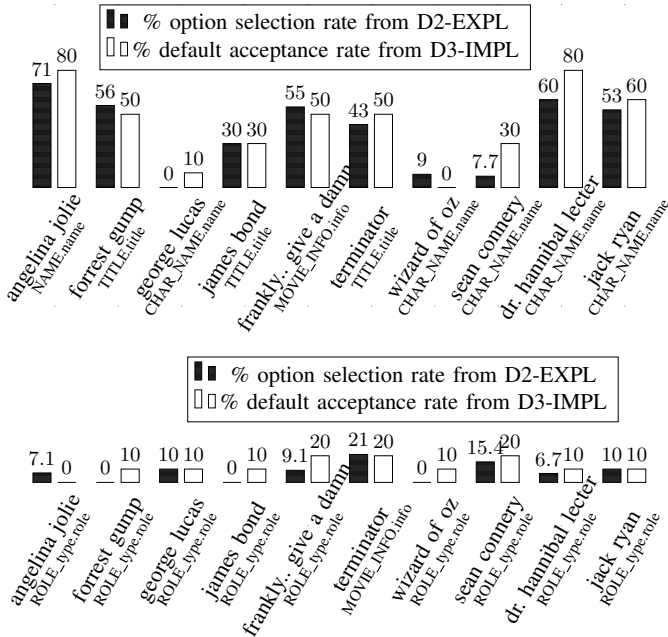


Fig. 8. Explicit and implicit choice selection rates, with a good default (top row) and bad default (bottom row). Even though there is considerable variability across terms, there is a clear lazy abandonment gap of about 5%.

when a different default option is used, we ran an additional experiment with the implicit feedback set up. Since D3 has been produced by presenting as default option the highest scored option from D1-INTR, such additional check would be most effective to stress ℓ if we now show D1’s lowest scored option as default. We rendered a dataset D3’ this way, as a variant of D3 where we knew the default answer was wrong.

Specifically, we instantiated a task to 10 new workers for each query term, and showed them such a ‘bad’ default. Fig. 8 (bottom) shows results based on D3’. We also see fairly close explicit and implicit feedback rates. For term ‘forrest gump,’ e.g., the ‘bad’ default was ROLE_TYPE.role, which obtained D2-score 0% and D3-score 10%. For this term we have an estimated laziness of $\ell = 10\%$. Over all the 62 terms, we have mean $\ell = 4.8 \pm 8\%$.

In sum, first, in both cases the rates from explicit (D2) and implicit feedback (D3 or D3’) are similar termwise. Second, the average laziness rates (resp., 5.1%, and 4.8%) are compatible across the two different settings. That is, if our crowd tasks have a reasonable design to simulate data retrieval interactions then the laziness parameter has empirical support to be set $\ell \approx 5\%$. On average, therefore, users have only a 5% bias towards just accepting the default choice whatever it is. In other words, implicit feedback is expected to work nearly as well as explicit feedback, which is exactly what we observed in the evaluation of §V.

VII. RELATED WORK

Machine learning models. Problems like ours are often addressed by learning a classifier from labeled training data, and then applying it to the unseen instances.¹⁰ This is a popular approach, when (i) the training data is really large and diverse enough to avoid overfitting; and (ii) no insight is available other

than the training data. Recall, however, that (i) D5 has only 62 examples (labeled queried terms). Having such low number is common since acquiring labeled data is expensive; and the interaction log for the queried terms population may form a ‘long tail,’ possibly making the real system log small termwise; also, (ii) D1-INTR is available with 60% P@1 accuracy; it is therefore a clearly useful information source to build upon.

So the IDR problem setting and how to make the best of its two sources is more challenging and does not suit standard supervised learning. That is why we have taken a more original approach for term classification.

Combination of experts. Many models combine forecasts from several human experts as independent information sources on uncertain events. They give their probability estimates, which would be compatible with relative frequencies obtained from multinomial counts like in a user interaction log. A popular approach to combine the individual forecasts is the so-called ‘linear opinion pooling.’ It assigns each forecast a weight that reflects the importance or quality of that expert. Recently, along these lines, more advanced techniques have been developed [22]. One important point of improvement is to consider ‘sharpness,’ which rewards how close to either 0 or 1 an estimate is (ibid.), somehow in line with our choice to use entropy as a measure. Yet their core measure of source quality is still ‘calibration’ (see, e.g., Brier scores).¹¹ Suppose an expert estimates 0.3 as the probability for a specific outcome, then she is best calibrated if that outcome is seen in 30% of the trials. Calibration is dependent on seen examples.

Consider a scenario where the intrinsic and extrinsic sources are calibrated given D1-INTR and D3-IMPL as training data, and D5-BENCH as testing data. By Fig. 4, D1 turns out not to be calibrated at all, while D3 is very well calibrated. Note the structural asymmetry. So these “experts” are given (resp.) very low and very high importance. For term ‘tom hanks,’ e.g., whose D1 scores and D3 log counts are shown in Example 1, our two “experts” would be given asymmetrical importance likewise. Now suppose the real system’s log gets noisy because a fan of the ‘Bamboo Shark’ movie is repeatedly querying for ‘tom hanks,’ the name of a character in this movie, rendering the important log source effectively “bad” for next queries on ‘tom hanks,’ the person’s name.

Without a more complex model, e.g., to give personalized results, requiring then even more expensive training with profiled users etc, there is no way calibration could help the system to downgrade the log source and upgrade the intrinsic source “experts” on the fly. Calibration can only distinguish source quality based on training data that is expensive to acquire, and yet becomes outdated: (1) as the database is updated with new records, properly captured by the ‘uncalibrated’ intrinsic model by the way; and (2) the interaction log is updated with real use, captured by the occasionally ‘uncalibrated’ entropy. Entropy is instead a unsupervised measure. It can be checked very fast every new interaction log update. For term logs that have high entropy, either due to genuine ambiguity (e.g., ‘terminator’) or due to unusual noise, Bsmooth will be conservative and give more weight to the intrinsic model.

The Dempster-Shafer theory of evidence [23] has been applied in related work to combine two DB-intrinsic scorings [24]. It is not fit to the interaction log scenario. The inferential

¹⁰Were the disambiguation response variable real-valued and not categorical then the problem would be one of regression, not classification [8].

¹¹http://en.wikipedia.org/wiki/Brier_score.

scheme (Dempster’s rule of combination) derives shared belief from multiple sources and ignores all the conflicting (non-shared) belief by a normalization factor. Zadeh gives an example of how counter-intuitive this is when beliefs should be rather integrated cumulatively [25]. Let doctors S_1, S_2 have beliefs for diagnosis on conditions (A) meningitis, (B) brain tumor, (C) concussion, viz., $S_1 = \{P(A) = .99, P(B) = .01\}$ and $S_2 = \{P(B) = .01, P(C) = .99\}$. Then $P(B) = 1$ is inferred [25]. The doctors agree only that brain tumor is very unlikely, but such a weak consensus is pushed through anyway. Now if we think of A, B, C as movies S_1, S_2 plan to see together then the inference finds the movie shared by their belief constraints. This shows that the matching between abstract framework and applied use case is really important.

Dempster-Shafer’s theory has applications of ‘truth finding’ given conflicting and/or absent information, when sources are independent — distributed, say, in time, space or perspective; e.g., in expert systems, question answering, data fusion, etc. The IDR use case, however, is distinguished by an asymmetry between the DB-intrinsic (model) vs. the DB-extrinsic (data) as we have extensively discussed, and seen in §V-B.

Interactive Data Retrieval. As mentioned in §II, we used crowdsourcing as a cost-effective means to simulate user interactions and get insight to system design beforehand [6]. By two different crowd task designs, we have acquired explicit and implicit feedback and have studied them, also considering their possible limitations to feed a learning layer in IDR systems. Although we take some inspiration from Interactive Information Retrieval (IIR) [26], we avoid complex models (e.g., [27]) to pursue and report instead a learning technique designed to be simple and very fast at query time. Note also that a comparison with complex models for aggregation of crowd answers is not required, since a direct measuring of its uncertainty has been enough to warrant high P@1 accuracy under a neat bayesian smoothing model tuned with LOGIT.

Regarding implicit feedback in particular, the concept of ‘good abandonment’ (GA) has been explored based on a large Google search log [2]. It indicates that GA (i.e., when the user’s information need is satisfied with no need to click on a result or refine the query) may be really a significant portion of abandoned sessions. Considering two modalities, PC and mobile, they point out that the latter has higher GA rates. As mobile queries tend to be more objective, the result snippets are often enough to satisfy an information need [2]. This adds to our own findings suggesting high GA rates in the IDR use case, as a database query answer (structured list of facts) may be precise enough not to require any browsing.

Imprecise DB queries. Some ad-hoc practices in the keyword search literature that have been criticized [28], [21], [3], and that we have strived not to incur in are:

- Existing scoring functions have become increasingly complex while the added value obscure [3]. We have studied and shown in detail the added value of implicit feedback as a DB-extrinsic scoring to be combined by Bsmooth with any existing DB-intrinsic scoring.
- Related work reports evaluation based on ad-hoc queries and databases—even arbitrary modification of schema, e.g., changing table and attribute names to match user queries better [3]. We evaluate Bsmooth on IMDb ‘as is’ in the benchmark with its own encoding.

- As mentioned, most related techniques rely on large DB-induced graphs and/or on auxiliary views for keywords, incurring in serious performance and maintenance issues. They may not apply to online databases on-the-fly [21]. Bsmooth in turn can be built at query time on top of any intrinsic model without noticeable time expense added.
- Most systems retrieve tuples in a one-shot response. We instead leverage on the potential of IDR with some interactions that are cheap to users. The learning layer we propose can be incorporated by any IDR system.

VIII. CONCLUSIONS

In this paper we have studied imprecise DB queries in the context of interactive data retrieval (IDR), where a little help is expected from users to improve the disambiguation of query terms. The potential of relying on a DB-extrinsic source in addition to the DB-intrinsic one is obvious, yet uncertainty management is not trivial. We have singled out bayesian smoothing as an adequate learning framework in the presence of noise and possibly small query logs, and seen several alternative weighting schemes that could have been built into the proposed Bsmooth algorithm to do it wisely.

In fact, by simulating user interactions we have found that a crowd of users can potentially provide highly accurate feedback into a query log, even when it is collected implicitly termwise and the workers have no knowledge of the DB internal structure. Here, one could think of storing feedback not only by term but also by query, by individual user, or even by time (sequential) leading to more complex models. Yet we have seen that none of these seem to be required to achieve highly accurate type disambiguation in IDR even given a small-size query term log.

We have opted to advance in depth using one DB instead of in breadth with multiple DB’s—the benchmark, e.g., covers Wikipedia and Mondial datasets as well [3]. Testing Bsmooth against these other datasets would be straightforward, if not for the cost of crowdsourcing the 50 queries (i.e., 50+ terms) for each DB over multiple settings to simulate user feedback. Our work, nonetheless, establishes a first direction that can be easily incorporated into off-the-shelf IDR systems.

The bottom line is that user feedback, even implicit, can greatly enhance the quality of disambiguation in IDR systems. We recommend the use of a LOGIT strategy, tuned with a regularity factor (H_D) of .75, to balance the intrinsic and extrinsic evidence. Algorithms 1–2 give the complete recipe.

REFERENCES

- [1] F. Li and H. V. Jagadish, “Constructing an interactive natural language interface for relational databases,” *PVLDB*, vol. 8, no. 1, pp. 73–84, 2015.
- [2] J. Li, S. Huffman, and A. Tokuda, “Good abandonment in mobile and pc internet search,” in *Proc. of ACM SIGIR*, 2009, pp. 43–50.
- [3] J. Coffman and A. C. Weaver, “An empirical performance evaluation of relational keyword search techniques,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 30–42, 2014.
- [4] —, “A framework for evaluating database keyword search strategies,” in *ACM Proc. of CIKM*, 2010, pp. 729–38.
- [5] A. Kittur, E. H. Chi, and B. Suh, “Crowdsourcing user studies with Mechanical Turk,” in *Proc. of ACM CHI*, 2008, pp. 453–6.
- [6] G. Zucco *et al.*, “Crowdsourcing interactions: using crowdsourcing for evaluating interactive information retrieval systems,” *Information Retrieval*, vol. 16, no. 2, pp. 267–305, 2013.

- [7] C. Cleverdon, *In Readings in Information Retrieval*. Morgan Kaufmann Publishers, 1997, ch. The Cranfield Tests on Endex Language Devices, pp. 47–59.
- [8] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.
- [9] C. R. Shalizi, *Advanced Data Analysis from an Elementary Point of View*. Cambridge University Press, 2015.
- [10] C. D. Manning *et al.*, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [11] G. Bhalotia *et al.*, “Keyword searching and browsing in databases using BANKS,” in *ICDE*, 2002, pp. 431–40.
- [12] V. Hristidis and Y. Papakonstantinou, “DISCOVER: Keyword search in relational databases,” in *VLDB*, 2002, pp. 670–81.
- [13] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, “Bidirectional expansion for keyword search on graph databases,” in *VLDB*, 2005, pp. 505–16.
- [14] V. Hristidis, L. Gravano, and Y. Papakonstantinou, “Efficient IR-style keyword search over relational databases,” in *VLDB*, 2003, pp. 850–61.
- [15] F. Liu, C. Yu, W. Meng, and A. Chowdhury, “Effective keyword search in relational databases,” in *SIGMOD*, 2006, pp. 563–74.
- [16] B. Ding *et al.*, “Finding top-k min-cost connected trees in databases,” in *IEEE ICDE*, 2007, pp. 836–45.
- [17] H. He, H. Wang, J. Yang, and P. S. Yu, “BLINKS: Ranked keyword searches on graphs,” in *SIGMOD*, 2007, pp. 305–16.
- [18] Y. Luo, X. Lin, W. Wang, and X. Zhou, “SPARK: Top-k keyword query in relational databases,” in *SIGMOD*, 2007, pp. 115–26.
- [19] G. Kasneci, M. Ramanath, M. Sozio, F. M. Suchanek, and G. Weikum, “STAR: Steiner-tree approximation in relationship graphs,” in *ICDE*, 2009, pp. 868–79.
- [20] J. Coffman and A. Weaver, “Structured data retrieval using cover density ranking,” in *KEYS*, 2010, pp. 1–6.
- [21] A. Baid *et al.*, “Toward scalable keyword search over relational data,” *PVLDB*, vol. 3, no. 1, pp. 140–9, 2010.
- [22] V. A. Satopää *et al.*, “Combining multiple probability predictions using a simple logit model,” *Int. J. Forecast.*, vol. 30, no. 2, pp. 344–56, 2014.
- [23] G. Shafer, *A Mathematical Theory of Evidence*. Princ. U. Press, 1976.
- [24] S. Bergamaschi *et al.*, “QUEST: a keyword search system for relational data based on semantic and machine learning techniques,” *PVLDB*, vol. 6, no. 12, pp. 1222–5, 2013.
- [25] L. Zadeh, “Book review: A mathematical theory of evidence,” *AI Magazine*, vol. 5, no. 3, pp. 81–3, 1984.
- [26] L. Azzopardi *et al.*, “Report on the SIGIR 2010 workshop on the simulation of interaction,” *ACM SIGIR Forum*, vol. 44, no. 2, pp. 35–47, 2010.
- [27] L. Azzopardi, “Modelling interaction with economic models of search,” in *ACM SIGIR*, 2014, pp. 3–12.
- [28] W. Webber, “Evaluating the effectiveness of keyword search,” *IEEE Data Eng. Bull.*, vol. 33, no. 1, pp. 54–9, 2010.

IX. APPENDIX

A. Proof of Lemma 1

Let $\vec{x}(x_1, \dots, x_k)$ and $\vec{y}(y_1, \dots, y_k)$ be two scoring vectors and $t \neq u$ two of their bins. Then a convex combination $\vec{z} = \vec{x} \oplus \vec{y}$ with $z_t = z_u$ exists and is unique if, only if one of these mutually exclusive conditions hold:

- (i) $(x_t - x_u)/(y_t - y_u) < 0$,
- (ii) $(x_t - x_u) = 0$ but $(y_t - y_u) \neq 0$,
- (iii) $(x_t - x_u) \neq 0$ but $(y_t - y_u) = 0$.

Proof: Let \vec{z} be defined $z_i = \omega_1 x_i + \omega_2 y_i$ for all bins i and some $\omega_1, \omega_2 \geq 0$ such that $\omega_1 + \omega_2 = 1$. Then, by Def. 2, we have $\vec{z} = \vec{x} \oplus \vec{y}$ and we can write $z_t = \omega_1 x_t + \omega_2 y_t$ and $z_u = \omega_1 x_u + \omega_2 y_u$. Now we are interested in existence and uniqueness conditions for $z_t = z_u$.

In fact, we can find $\omega_1, \omega_2 \geq 0$ to satisfy both $\omega_1 + \omega_2 = 1$ and $\omega_1 x_t + \omega_2 y_t = \omega_1 x_u + \omega_2 y_u$ over constants $x_t, y_t, x_u, y_u \geq 0$. That is, our problem reduces to the solution set of the following linear system for $\omega_1, \omega_2 \geq 0$:

$$\begin{cases} \omega_1 + \omega_2 = 1 \\ (x_t - x_u)\omega_1 + (y_t - y_u)\omega_2 = 0 \end{cases}$$

We start with the ‘only if’ statement. That is, we must show that if $(x_t - x_u)/(y_t - y_u) > 0$ or $(x_t - x_u) = (y_t - y_u) = 0$ then a solution either cannot exist or is not unique.

In fact, suppose by contradiction that $(x_t - x_u)/(y_t - y_u) > 0$. By basic manipulation over the second equation we have $\omega_1/\omega_2 = -(y_t - y_u)/(x_t - x_u)$. Since we have assumed $(x_t - x_u)/(y_t - y_u) > 0$, then it must be the case that $\omega_1/\omega_2 < 0$. But $\omega_1, \omega_2 \geq 0$. \nexists . Now rather if $(x_t - x_u) = (y_t - y_u) = 0$, then the system is indeterminate—underconstrained to just $\omega_1 + \omega_2 = 1$. That is, a solution exists but is not unique.

It remains to show the ‘if’ statement. In fact, if $(x_t - x_u) = 0$ but $(y_t - y_u) \neq 0$, then we easily infer $\omega_2 = 0$, $\omega_1 = 1$ and condition (ii) is shown. A symmetric argument works for the converse to cover condition (iii). Thus in both cases the solution exists and is unique.

Now finally, recall that $\omega_1/\omega_2 = -(y_t - y_u)/(x_t - x_u)$ and let $\lambda = -(y_t - y_u)/(x_t - x_u)$. Then $\omega_1/\omega_2 = \lambda$ and by condition (i) we must have $\lambda > 0$. That is, we have both $\omega_1/\omega_2 = \lambda$ and $\omega_1 + \omega_2 = 1$ for $\lambda > 0$ —which leads to $\omega_2 = 1/(\lambda + 1)$ and $\omega_1 = \lambda/(\lambda + 1)$, where $\lambda > 0$ and depends only on the constants x_t, x_u, y_t, y_u . Therefore the solution exists and is unique. ■

B. Proof of Theorem 1

Let $\vec{x} = (x_1, \dots, x_k)$ and $\vec{y} = (y_1, \dots, y_k)$ be two scoring vectors with top bins $t \in [\vec{x}]$ and $u \in [\vec{y}]$. If these are disagreed top bins $t \asymp u$ of \vec{x} and \vec{y} , then a convex combination $\vec{z} = \vec{x} \oplus \vec{y}$ with both $z_t = z_u$ exists and is unique.

Proof: The proof is straightforward from Lemma 1 and Defs. 3–4. Note that if $t \asymp u$, then by Def. 4 we have $t \in [\vec{x}]$ and $u \in [\vec{y}]$ but either $t \notin [\vec{y}]$ or $u \notin [\vec{x}]$. We start with condition $t \notin [\vec{y}]$.

If $t \notin [\vec{y}]$, then by Def. 3 there must be some bin $s \neq t$ such that $y_t < y_s$. Also, since by assumption u is a top bin of \vec{y} we have $y_s \leq y_u$. Thus we must have $y_t < y_u$ and, therefore, $y_t - y_u < 0$. Now by Lemma 1.(i-ii), we know that $\vec{z} = \vec{x} \oplus \vec{y}$ with both $z_t = z_u$ exists and is unique if, only if $x_t - x_u \geq 0$. That is, when $x_t \geq x_u$. In fact, by assumption $t \in [\vec{x}]$ then, again by Def. 3, that must be the case.

The argument for condition $u \notin [\vec{x}]$ is analogous. ■

C. Empirical Validation of Fact 1

Tables III, IV present the empirical validation for Fact 1. Some relevant observations are:

- For each prior \vec{x}_i the lack of fit for the linear profile (Table III) is always significantly larger than the lack of fit for the LOGIT profile (Table IV).
- The optimal parameters β_0, β for the linear profile (Table III) are always close to the specific profile LINEAR, with $\beta_0 = \beta = 1$, as introduced in Remark 4 and used in the evaluation of §V.
- The lack of fit of LOGIT (Table IV) is always very small.

TABLE III. Optimal parameters β_0, β for the linear profile, where $P = \{(0, 1), (1, 0), (.75, \bar{\omega}_2)\}$ and $\bar{\omega}_2$ varies row-wise.

Linear profile				
prior	$(D, \bar{\omega}_2)$	β_0	β	lack of fit
\bar{x}_6	(.75, .637)	1.059	0.880	9.253E-2
\bar{x}_5	(.75, .584)	1.051	0.897	6.897E-2
\bar{x}_4	(.75, .513)	0.985	1.048	1.036E-1
\bar{x}_3	(.75, .412)	1.025	0.950	1.640E-2
\bar{x}_2	(.75, .260)	1.002	0.997	7.012E-5
\bar{x}_1	(.75, .203)	0.993	1.014	1.317E-3
\bar{x}_0	(.75, .034)	0.967	1.066	2.847E-2

TABLE IV. Optimal parameters β_0, β for LOGIT₃, with $|P| = 3$ where $P = \{(0, 1), (1, 0), (.75, \bar{\omega}_2)\}$ and $\bar{\omega}_2$ varies row-wise.

LOGIT ₃				
prior	$(D, \bar{\omega}_2)$	β_0	β	lack of fit
\bar{x}_6	(.75, .637)	244.559	324.925	6.989E-11
\bar{x}_5	(.75, .584)	253.127	336.633	1.813E-11
\bar{x}_4	(.75, .513)	261.633	348.343	2.653E-11
\bar{x}_3	(.75, .412)	181.671	242.399	7.216E-10
\bar{x}_2	(.75, .260)	162.432	217.703	4.839E-10
\bar{x}_1	(.75, .203)	050.270	068.766	9.709E-13
\bar{x}_0	(.75, .034)	138.507	188.908	3.469E-11

TABLE V. Tuned parameters β_0, β for the LOGIT₄ profile, with $|P| = 4$ where $P' = \{(0, 1), (1, 0), (.75, \bar{\omega}_2), (.80, \bar{\omega}'_2)\}$. The setting for \bar{x}_1 is the one used in our evaluation in §V.

LOGIT ₄ — FINAL TUNED PARAMETERS				
prior	$(D, \bar{\omega}_2)$	β_0	β	lack of fit
\bar{x}_6	(.75, .637)	09.628	12.272	1.597E-2
\bar{x}_5	(.75, .584)	11.065	14.094	4.972E-3
\bar{x}_4	(.75, .513)	10.018	13.119	4.099E-3
\bar{x}_3	(.75, .412)	09.009	12.372	2.131E-3
\bar{x}_2	(.75, .260)	10.854	15.841	5.074E-5
\bar{x}_1	(.75, .203)	19.654	27.994	1.110E-7
\bar{x}_0	(.75, .034)	842.59	1126.5	4.84E-10

D. Final Tuned Parameters for LOGIT

Table V presents the final tuned parameters for LOGIT₄, with $|P| = 4$ where $P' = \{(0, 1), (1, 0), (.75, \bar{\omega}_2), (.80, \bar{\omega}'_2)\}$. Fig. 9 shows the resulting curve profiles. We observe:

- The nearly constant $\bar{\omega}_2$ for prior \bar{x}_0 near $H_D = .75$ in Fig. 2 leaves no room for smoothness in its corresponding profile in Fig. 9, whose form is step-like.
- For the other priors we can have smoother profiles with a slope that is roughly the same.

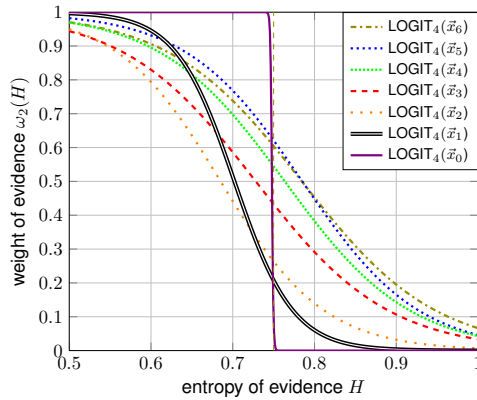


Fig. 9. Tuned curve profiles for LOGIT₄, with $|P| = 4$ where $P' = \{(0, 1), (1, 0), (.75, \bar{\omega}_2), (.80, \bar{\omega}'_2)\}$. The setting for \bar{x}_1 (double line) is the one used in our evaluation in §V.